**Back Yard Adventure Maintaining a Systems Development Project**

Morrese D. Morrison

Pierce College

Computer Science 302: System Analysis & Design

Study.Com

October 11th, 2023

## Back Yard Adventures Maintaining a Systems Development Project

Maintenance is often an overlooked aspect of software development and implementation. The general consensus is if it is not broken, then do not fix it. This rhetoric may be applicable with the mundane things in our everyday lives, but it does not and should not apply to software. Software of course is not physically real, you cannot see it or touch it, therefore individuals tend to believe that maintenance is not required since it is not a physical entity such as a car for example. If maintenance on a car is not preformed regularly, the performance of the car will degrade, and it will eventually become inoperable. The same thing can happen with software or systems.

Much like that car, if software is not maintained or configured to keep up with the needs of a its user, the use for that software will degrade to a point of inoperability. To avoid this, software systems need to be maintained and configured in order to sustain the same level of performance and to increase its overall function to accommodate the users possible evolving needs. These needs may include the ability to store records with different attributes attached to them. Or automated process that free up previously manual tasks. Or simple maintenance such as confirming that the code is secure from vulnerabilities and bottlenecks. Whatever it is, maintenance is an activity that should be performed on a daily basis to get the best of a software or system.

**Maintenance Forms & Activities**

There are several forms of software maintenance that can be performed when applicable. The main four software maintenance forms include corrective maintenance, adaptative maintenance, perfective maintenance, and preventative maintenance. All of these maintenance forms involve what their name implies, however we will take a closer look on how they are implemented in real time on software or systems. This is of course so we can also gain a greater understanding of what possibly will be needed for Back Yard Adventures as well. But before we dive into these four forms, we will discuss what software or system maintenance is at its core.

“The maintenance phase of the SDLC occurs after the product is in full operation. Maintenance of software can include software upgrades, repairs, and fixes of the software if it breaks” (Ransom, n.d).

So, within this maintenance phase, there are activities that are usually implemented in order to maintain the system. These activities include identifying the problem, analyzing what can be done to fix and mitigate the problem, designing and programing a code logic update, testing that fix, then implementing that fix into production. To explain these four maintenance activities, we will apply them to an example. Let’s say for example a telecommunications company has a new in-house software program that handles the production of employees, invoicing, payroll, revenue, and expenses. Within this company there are several different offices that preform the same type of work for a customer, but because of their respective area, certain rates in regard to production will be different depending on the scope of work.

One day, the development team recognizes an issue of the with the program when generating various profit reports. When generating the reports, the reports window will either crash or continue to spin and load no data, creating a blank report. These reports are crucial as the president, vice presidents and controller utilize these reports often to see what region has the most profit. These reports are also disseminated with shareholders and board members of that company as well. Once this issue is identified by the development team, they begin to look at the development environment of this program to see if they can recreate the issue. From the development environment, they indeed can replicate this issue.

There next step is to analyze what they can do to fix the issue. So, they open their IDE and look at the SQL stored procedure that is responsible for these reports. After analyzing the store procedure, they noticed that their SQL logic is not fully joining tables when executed. As a result, the code is creating a loop and eventually times out. They record this issue and immediately begin designing a solution. After about an hour, they have designed and reconstructed the stored procedure to fully join the intended table. They first tested the procedure straight from SQL server manager. They then utilized the program to call the procedure to see if it will work. Their newly reconstructed procedure work. After testing a bit more, they created an RFC in order to document the change. They then pushed the RFC through, and the change was made in the production environment. After the change, management was able to run all of their reports with no issue.

These steps are very effective if implemented correctly as seen in the example. These steps are also involved in a way with the four main maintenance forms we mentioned earlier as well. Again, these four maintenance forms are corrective maintenance, adaptive maintenance, perfective maintenance, and preventative maintenance. Corrective maintenance “is what one would typically associate with the maintenance of any kind” (*The 4 types of Software Maintenance & How They Help. N.d)*. This is a rather broad explanation but corrective maintenance deals with anything that needs to be corrected with the software. Whether that is updating API keys or simply changing the font so text can be readable on the page. These bugs can be reported to the development team, or the development team may spot them in advance.

Adaptive maintenance is “when the environment of your software changes” (*The 4 types of Software Maintenance & How They Help. N.d*). Now this usually is dealing with the services that interact with the software, where the software is hosted and its dependencies. An example of this would be a change in the production environment where the code is hosted, perhaps new security measures are instituted to prevent easy access to the source code and its dependencies. Or it can be an authentication change with a microservice that is involved with the software. Adaptive maintenance deals with “adapting” that software to a new or altered environment.

Perfective maintenance “focuses on the evolution of requirements and features that exist in your system” (*The 4 types of Software Maintenance & How They Help. N.d).* This form of maintenance is usually the most used in a business environment. Reason being is that often times, with an enterprise application, new business requirements are brought to the table from various stakeholders within the company. They may desire an alteration to the various reports that they run, including new fields to display new information for their analysis once they run the reports. Or a completely new function may need to be added to the enterprise software to complete a certain task. These requests are usually brought to the IT team. The IT team then will evaluate to see if the request is feasible and will not cause an issue later on. If it does not, then the IT team will begin to code, test, and implement the change in the test environment first, then implement in the production environment.

Preventative maintenance “helps to make changes and adaptations to your software so that it can work for a longer period of time” (*The 4 types of Software Maintenance & How They Help. N.d).* The primary focus is to prevent “deterioration of your software as it continues to adapt and change” (*The 4 types of Software Maintenance & How They Help. N.d).* Now what does that mean exactly? An example would be to avoid and eliminate any technical debt within the source code. This includes unused modules of codes, functions and algorithms that take a considerable long time to complete when processing data. Preventing and outright eliminating these occurrences will create less issues in the long run as new code is added. This new code will not have to be integrated with those negative occurrences, ultimately eliminating the future occurrence of future technical debt.

Technical debt also Segway’s into maintenance costs as a whole. “The cost involved in software maintenance are due to multiple factors and vary depending on the specific situation” (*What is a software maintenance process? 4 types of software maintenance*. N.d).

Whether you are building new software or maintaining old software, there are costs associated with the maintenance. But there some strategies to mitigate the cost a bit. One strategy includes utilizing functional programming. This style of programming is based on mathematical functions and is great for calculating sums, averages, differences, and percentages. This may alleviate the total lines of code needed for the software to function properly. Rather depending on user input instead to generate the desired data outcome. Another strategy might be to increase testing standards so bugs can be caught at a greater rate before released into production.

In addition to all of these forms and activities, there is one final activity that is often overlooked but just as imperative. And that is knowledge retention. This mainly includes creating and storing documentation about the software and its various dependencies. These documents should be organized, accurate and updated regularly. On top of that these documents should be easily accessible so they can be referenced at any time. As good as we may be with storing information in our heads, we will forget key aspects of a module, or why we coded a feature a certain way. This is where documentation can be utilized to speed up development time or maintenance time, further reducing cost and increasing overall output. This includes procedures as well, in the event of an emergency. Often referred to as “emergency plans”. What if the production environment is down? What if the repository has been compromised by external forces? Having documentation in regard to policies and procedures will prevent a bad situation from becoming worse. These policies and procedures should be discussed and reviewed regularly to ensure all are aware of what to do in the case of an emergency.

**Back Yard Adventures Forward & Future Maintenance Plan**

Now that we are well familiarized with the various maintenance forms and activities, a maintenance plan should be constructed for Back Yard Adventures. A maintenance plan that considers all possible issues and future requests. Before the general maintenance activities are outlined, it is imperative that we mention the ability for software reengineering for the system. Software reengineering deals with the “examination of a system to reconstitute it in a new form” (Software re-reengineering 2022, December 15). This basically means molding and shaping an existing system into a completely brand-new system. This is often carried out when a huge shift in requirements is needed that the current system cannot provide, or the simple fact that the old system is outdated and cannot be used. If this ever where the case, the system can go through a program restructuring. With all of the data being fed to the MySQL data base, this provides an opportunity for that data to flow in and out of the database in a myriad of ways depending on the approach that will be used. This essentially creates one point of contact where the data is stored, allowing forward engineering to take place to accommodate new business requirements and data points. However, program restructuring should be strongly analyzed with both the pros and cons in hand before this endeavor is taken.

The maintenance plan of the Back Yard Adventures System should be completed as followed. The use of all four maintenance forms should be utilized when applicable. These four forms of course include corrective maintenance, adaptive maintenance, perfective maintenance, and preventative maintenance. Before any changes or maintenance activities are implemented in production, an RFC should be submitted by the programmer detailing the change. An RFC stands for request for comments, which details a proposed fixed to the system. Now this change can be to correct a bug or to implement a new requirement within the system. This RFC should be and signed off by both Harry and Shawn. Even if Harry or Shawn requests the change themselves, an RFC should still be submitted, reviewed, and approved by both before change implementation. This creates an audit trail of changes as well as confirming if the change is actually required. Emergency RFCs encompasses a system outage or bug that needs to be fixed immediately. In this case, an Emergency RFC may be submitted after the fix is implemented in order to keep operations afloat. The RFC should still be signed off and filled correctly for future reference.

## Conclusion

Maintenance plays a crucial part in the software development life cycle. As such it should be emphasized as a requirement going forward with Back Yard Adventures current system. Preforming routine maintenance and checks will ensure that the system preforms up to the required standard that has been set. Sticking to the maintenance plan will also allow for customization of ingesting certain data points and the overall function of the system. As long as the four maintenance models and activities are followed, the upkeep and performance of Back Yard Adventures System will be organized and accurate.

## References

GeeksforGeeks. (2022, December 15). *Software re-engineering*. GeeksforGeeks.

[https://www.geeksforgeeks.org/software-re-engineering/](GeeksforGeeks.%20(2022,%20December%2015).%20Software%20re-engineering.%20GeeksforGeeks.%20https:/www.geeksforgeeks.org/software-re-engineering)

Ransom, N. (n.d) *Maintenance Phase in SDLC*

<https://study.com/academy/lesson/maintenance-phase-in-sdlc.html>

*The 4 types of Software Maintenance & How They Help: Cast Software*. The 4 Types Of Software Maintenance & How They Help | CAST Software. (n.d.). <https://www.castsoftware.com/glossary/four-types-of-software-maintenance-how-they-help-your-organization-preventive-perfective-adaptive-corrective>

*What is a software maintenance process? 4 types of software maintenance*. (n.d.).

<https://cpl.thalesgroup.com/software-monetization/four-types-of-software-maintenance>